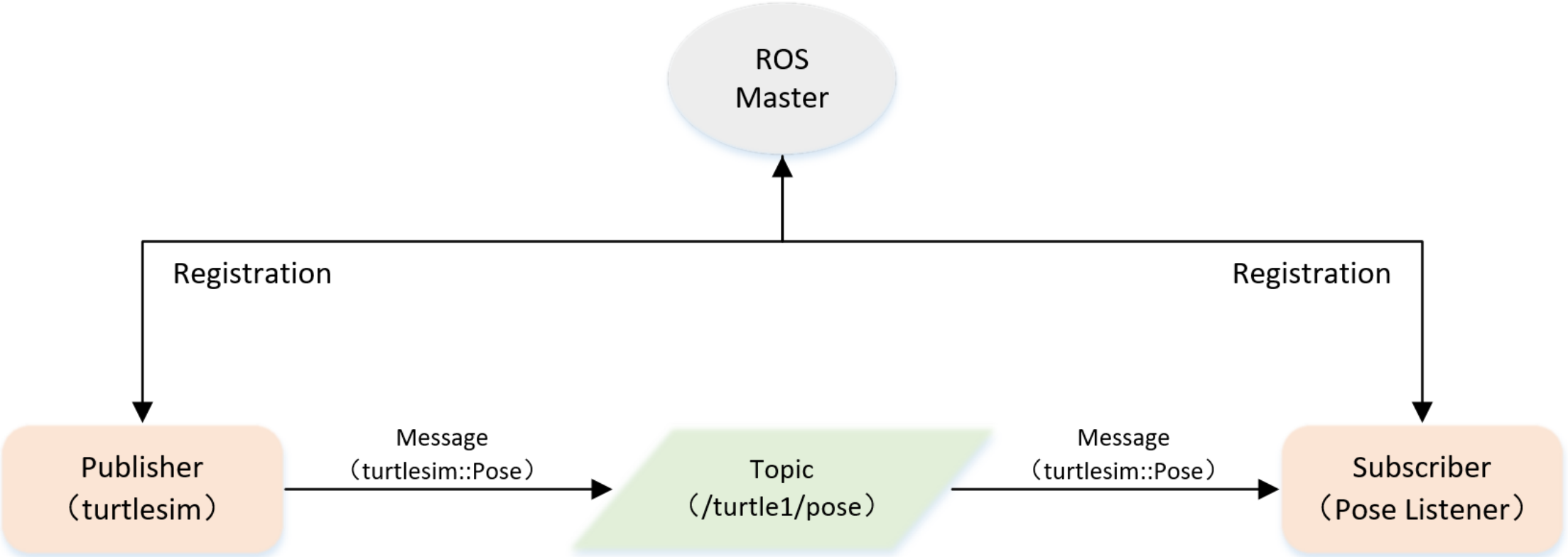
古月居 GYH.AI ROS

ROS入门
21讲

11.订阅者Subscriber的编程实现

主讲人：古月

话题模型



话题模型（发布/订阅）

创建订阅者代码（C++）

```cpp
/**
 * 该例程将订阅/turtle1/pose话题，消息类型turtlesim::Pose
 */

#include <ros/ros.h>
#include "turtlesim/Pose.h"

// 接收到订阅的消息后，会进入消息回调函数
void poseCallback(const turtlesim::Pose::ConstPtr& msg)
{
    // 将接收到的消息打印出来
    ROS_INFO("Turtle pose: x:%0.6f, y:%0.6f", msg->x, msg->y);
}

int main(int argc, char **argv)
{
    // 初始化ROS节点
    ros::init(argc, argv, "pose_subscriber");

    // 创建节点句柄
    ros::NodeHandle n;

    // 创建一个Subscriber，订阅名为/turtle1/pose的topic，注册回调函数poseCallback
    ros::Subscriber pose_sub = n.subscribe("/turtle1/pose", 10, poseCallback);

    // 循环等待回调函数
    ros::spin();

    return 0;
}
```

pose_subscriber.cpp

如何实现一个订阅者

- 初始化ROS节点；
- 订阅需要的话题；
- 循环等待话题消息，接收到消息后进入回调函数；
- 在回调函数中完成消息处理。

配置订阅者代码编译规则

```
## Declare a C++ executable
## With catkin_make all packages are built within a single CMake context
## The recommended prefix ensures that target names across packages don't collide
# add_executable(${PROJECT_NAME}_node src/learning_topic_node.cpp)

## Specify libraries to link a library or executable target against
# target_link_libraries(${PROJECT_NAME}_node
#   ${catkin_LIBRARIES}
# )

add_executable(velocity_publisher src/velocity_publisher.cpp)
target_link_libraries(velocity_publisher ${catkin_LIBRARIES})


add_executable(pose_subscriber src/pose_subscriber.cpp)
target_link_libraries(pose_subscriber ${catkin_LIBRARIES})
```

如何配置CMakeLists.txt中的编译规则

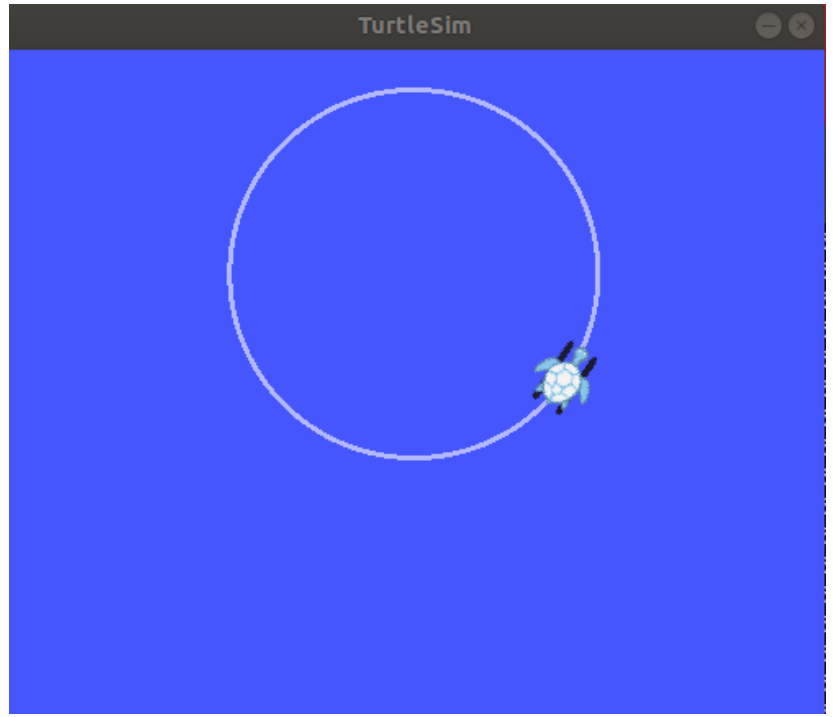
- 设置需要编译的代码和生成的可执行文件；
- 设置链接库；

```
add_executable(pose_subscriber src/pose_subscriber.cpp)
target_link_libraries(pose_subscriber ${catkin_LIBRARIES})
```

CMakeLists.txt

编译并运行订阅者

```
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
$ roscore
$ rosrun turtlesim turtlesim_node
$ rosrun learning_topic velocity_publisher
```




```
hcx@hcx-vpc:~/catkin_ws$ rosrun learning_topic pose_subscriber
[ INFO] [1562211557.322259871]: Turtle pose: x:6.389005, y:10.396028
[ INFO] [1562211557.339097278]: Turtle pose: x:6.381475, y:10.398730
[ INFO] [1562211557.354512018]: Turtle pose: x:6.373938, y:10.401410
[ INFO] [1562211557.370549572]: Turtle pose: x:6.366391, y:10.404065
[ INFO] [1562211557.387085434]: Turtle pose: x:6.358836, y:10.406695
[ INFO] [1562211557.402710847]: Turtle pose: x:6.351273, y:10.409303
[ INFO] [1562211557.418887039]: Turtle pose: x:6.343701, y:10.411885
[ INFO] [1562211557.434469988]: Turtle pose: x:6.336121, y:10.414443
[ INFO] [1562211557.450210135]: Turtle pose: x:6.328533, y:10.416977
[ INFO] [1562211557.465994903]: Turtle pose: x:6.320937, y:10.419487
[ INFO] [1562211557.482173454]: Turtle pose: x:6.313333, y:10.421972
```

创建订阅者代码（Python）

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# 该例程将订阅/turtle1/pose话题，消息类型turtlesim::Pose

import rospy
from turtlesim.msg import Pose

def poseCallback(msg):
    rospy.loginfo("Turtle pose: x:%0.6f, y:%0.6f", msg.x, msg.y)

def pose_subscriber():
    # ROS节点初始化
    rospy.init_node('pose_subscriber', anonymous=True)

    # 创建一个Subscriber，订阅名为/turtle1/pose的topic，注册回调函数poseCallback
    rospy.Subscriber("/turtle1/pose", Pose, poseCallback)

    # 循环等待回调函数
    rospy.spin()

if __name__ == '__main__':
    pose_subscriber()
```

pose_subscriber.py

如何实现一个订阅者

- 初始化ROS节点；
- 订阅需要的话题；
- 循环等待话题消息，接收到消息后进入回调函数；
- 在回调函数中完成消息处理。

感谢观看

怕什么真理无穷，进一寸有一寸的欢喜

更多精彩，欢迎关注

古月居

古月学院

ROS入门
21讲

古月居 搜一下